Goals

• See how to use ChronoSync in an application
• See how to use the sync API of the NDN client library
Overview

• Review the client library ChronoSync support
• Explore an example ChronoSync application: FireChat
• Deep dive: Follow code to “send” a chat message using ChronoSync
ChronoSync2013

- Implemented in NDN client libraries for NDN-CPP, PyNDN, NDN-JS, jNDN
- Main functionality:
  - Maintain the latest “sequence number” for each user
  - Publish a new sequence number from me
  - Notify on a new sequence number from another user
- Separate application-specific messages based on the “sequence number”

<table>
<thead>
<tr>
<th>Alice’sPrefix</th>
<th>Bob’sPrefix</th>
<th>Ted’sPrefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>8</td>
<td>35</td>
</tr>
</tbody>
</table>

2015/9/30
Names

• Sync data name to represent the dataset status:
  • Multicast_prefix/ApplicationName/digest
  • /ndn/multicast/CHAT/CHANNEL/tutorial/
d04f8183fe685488a5ba6763869fc93e19a6c5e5038518e3e5818516b307bba6

• Application data name:
  • Participant_prefix/ApplicationName/msg_seq
  • /ndn/org/icn/USER/bob%40ucla.edu/CHAT/CHANNEL/tutorial/SESSION/1442864410/3
ChronoSync-Based App Design Overview

ChronoSync-Based App

App-specific logic and state storage (chat messages, file revisions, etc.)

ChronoSync instance

Digest Tree

Sync Interests

/mcast/app/instance

Sync Data

/a1324adfa

Recovery Interests

Recovery Data

Notifcations about new data source statuses

Interest/Data to retrieve application data, e.g., inferred from Sync Data

Interest

Data
ChronoSync-Based App Design Overview

ChronoSync-based App

- App-specific logic and state storage (chat messages, file revisions, etc.)

ChronoSync instance

- Digest Tree
- Digest Log (optional)

Interest/Data to retrieve application data, e.g., inferred from Sync Data

Notifications about new data source statuses

Sync Interests
- /mcast/app/instance
- /a1324adfa

Sync Data

Recovery Interests

Recovery Data
- /mcast/app/instance
- /recovery/a1324adfa

Interest
FireChat

- Use ChronoSync2013 with assumptions for our chat app
- Inspired by the simple interface of Firebase [http://firebase.com](http://firebase.com)
- Peer-to-peer
- new FireChat(screenName, username, chatRoom, ...);
- Simple methods and JavaScript callbacks
- [https://github.com/zhehaowang/icn-tutorial-app](https://github.com/zhehaowang/icn-tutorial-app)
FireChat assumptions

• Stuff we won’t have to worry about…
• Connect to an NFD host at UCLA over WebSockets
• Fixed name prefix for chat messages
• JSON for the chat message content
• User keys stored locally in-browser with IndexedDB
• User certificates issued by an automated authority at UCLA
• Hard-wired certification trust root for the automated authority
FireChat application design

browser UI

FireChat class

send, leave, …
onChatData, …
publishNextSeqNo, expressInterest, …
onReceivedSyncState, onData, …

NDN-JS

ChronoSync2013 class

browser IndexedDB

WebSockets

local storage

network
Create session

var username = "alice@ucla.edu";
var screenName = "alice";
var chatroom = "tutorial";
var chronoChat = new FireChat(screenName, username, chatroom,
onChatData, onUserLeave, onUserJoin, updateRoster,
onChatDataVerified);

• If needed, generate user keys
• Register with NFD to receive interests
• Join the chat room
• Sync to the latest chat message “sequence number” from other users
• Set up “heartbeat” timer - missed heartbeat from another means “leave”
Send chat messages

var message = "Funny & true pic <img ... />";
chronoChat.send(message);

- Get my next chat message sequence number, update the digest tree
- Reply to sync messages with the new sequence number
- Put the chat message in the in-memory log, ready to reply to interests
- Put the chat message in persistent storage for “recovery” from other users
- message is HTML, suitable for <div></div>
- Can link to images or content (not part of the chat message)
- (deep dive follows)
Deep dive: Follow code for send

Browser UI

sendMessageClick

send

messageCacheAppend

msgCache

publishNextSequenceNo

onInterest

putData

FireChat

NDN-JS
Deep dive: page.js click SUBMIT

```javascript
$("#chatBtn").click(function() { sendMessageClick(); });

function sendMessageClick() {
  var chatMsg = $('#chatTextInput').val();
  var escaped_msg = $('<div/>').text(chatMsg).html();
  chronoChat.send(escaped_msg);
  ...
}
```

Diagram:
```
Browser UI
sendMessageClick
FireChat
send
```
Deep dive: FireChat.send

FireChat.prototype.send = function(msg) {
    ...
    this.messageCacheAppend("CHAT", msg);
    this.onChatData(this.screenName, new Date().getTime(), msg);
};

• messageCacheAppend does most of the work (next slide)
• Call onChatData so the application also displays its own chat messages
Deep dive: FireChat.messageCacheAppend

FireChat.prototype.messageCacheAppend = function(messageType, message) {
    this.sync.publishNextSequenceNo();
    var content = new FireChat.ChatMessage
        (this.sync.getSequenceNo(), this.username, this.screenName, messageType, message, new Date().getTime());
    this.msgCache.push(content);
    // Also put the message in the persistent chat storage.
    ...
};

• Publish the next sequence number
• Save the ChatMessage JSON object and wait for interests from other users
Deep dive: FireChat.onInterest

FireChat.prototype.onInterest = function(prefix, interest, face, ...) {
    var seq = parseInt(interest.getName().get(-1).toEscapedString());
    var chatMessage = findChatMessage(this.msgCache, seq);
    var data = new Data(interest.getName());
    data.setContent(chatMessage.encode());
    this.keyChain.sign
        (data, this.certificateName, function() {
            face.putData(data);
        });
};

- /ndn/org/icn/USER/alice%40ucla.edu/CHAT/CHANNEL/tutorial/SESSION/1442864410/5
- keyChain.sign explained later
- face.putData sends the data packet to the face of the incoming interest
Deep dive: Chat message content

{"seqNo": 5,
"fromUsername": "alice@ucla.edu",
"fromScreenName": "alice",
"msgType": "CHAT",
"timestamp": 1442932978694,
"data": "funny & true",
"to": ""}
DeepDive: ChronoSync2013.publishNextSequenceNo

`ChronoSync2013.prototype.publishNextSequenceNo = function() {
  this.usrseq++;  
  var message = makeSyncMessage  
    (this.applicationDataPrefixUri, this.usrseq, this.session);
  this.broadcastSyncState(this.digest_tree.getRoot(), message);  
  this.digest_tree.update(message); // (actual code is more detailed)  
  var interest = new Interest(this.applicationBroadcastPrefix);  
  interest.getName().append(this.digest_tree.getRoot());  
  this.face.expressInterest(interest, this.onData.bind(this), ...);
};

• broadcastSyncState will reply to interests for the previous digest with the new usrseq
• Express interest for next digest root: /ndn/multicast/CHAT/CHANNEL/tutorial/d04f8183fe685488a5ba6763869fc93e19a6c5e5038518e3e5818516b307bba6
Receive join and leave notifications

```javascript
function onUserJoin(from, time, msg, verified) { ... }
function onUserLeave(from, time, msg, verified) { ... }
```

- Notifies another user’s screen name who joins or leaves
- Call the callback once for each join or leave
Receive chat messages

```javascript
function onChatData(from, time, msg, verified) { ... }
```

- Notifies another user’s screen name and chat message
- Call the callback once for each message
- The message is HTML, suitable for `<div></div>`
- `/ndn/org/icn/USER/bob%40ucla.edu/CHAT/CHANNEL/tutorial/SESSION/1442864410/3`
Leave

FireChat.prototype.leave = function() { ... };

- Send the leave message
- Stop receiving other user’s messages
Putting it together

- **index.html:**
  - Include ndn.min.js, page.js, fire-chat.js and indexeddb-storage.js
  - HTML for the chat page text areas, buttons, etc.
  - HTML for the initial prompt for email and screen name: `<div id="email-dialog">`

- **page.js:**
  - `$("#email-dialog").close`: Call `startFireChat()`.
  - `startFireChat()`: new `FireChat` (screenName, username, chatroom, `onChatData`, onUserLeave, onUserJoin, updateRoster, …);
  - `onChatData`, onUserLeave, onUserJoin, updateRoster: Display messages

- **fire-chat.js:**
  - The `FireChat` class

- **indexeddb-storage.js:**
  - The IndexedDbChatStorage class, called from `FireChat.messageCacheAppend`
Goals recap

• See how to use ChronoSync in an application
• See how to use the sync API of the NDN client library